In the Claims

1. (Original) A method for reporting data from a legacy computer system using Extensible Markup Language, the method comprising:

generating a model of the legacy computer system;

mapping the model of the legacy computer system to an Extensible Markup Language schema; and

automatically modifying one or more applications of the legacy computer system, the modified application operable to output data written using a Document Object Model from the legacy computer system in Extensible Markup Language.

2. (Original) The method of Claim 1 wherein automatically modifying one or more applications further comprises:

providing the legacy computer system with a writer engine, the writer engine having the Extensible Markup Language Schema loaded as a data file; and

calling the writer engine with the modified applications, the writer engine populating the Document Object Model according to the Extensible Markup Language schema by building a Document Object Model instance with one or more contexts.

3. (Previously presented) The method of Claim 2 further comprising:

applying one or more XSLT stylesheets to restructure the Document Object Model instance for outputting data in a predetermined format.

4. (Original) A system for reporting data from a legacy computer system in an Extensible Markup Language format, the system comprising:

a modeling engine in communication with the legacy computer system, the modeling engine operable to generate a model of reported data written by an application residing on the legacy computer system;

a mapping engine in communication with the modeling engine, the mapping engine operable to generate a modification specification by mapping the model to an Extensible Markup Language schema; and

a code generation engine in communication with the mapping engine and the legacy computer system, the code generation engine operable to modify legacy computer system application code to directly output data from a Document Object Model as Extensible Markup Language.

5. (Original) The system of Claim 4 further comprising:

a context table associated with the legacy computer system, the context table providing the Extensible Markup Language schema to the legacy computer system; and

a writer engine loaded on the legacy computer system and having the Extensible Markup Language schema stored as a data file, the writer engine communicating with the modified legacy computer system applications to buffer data in plural contexts within a Document Object Model for output as Extensible Markup Language.

6. (Original) The system of Claim 5 wherein the writer engine is coded in the computer language of the legacy computer system.

7. (Original) A method for outputting data from an application running on a computer system, the data output as Extensible Markup Language, the method comprising:

establishing a relationship of the output data and one or more Extensible Markup Language Document Object Model contexts;

building a Document Object Model instance with the one or more contexts; and

outputting the data from the Document Object Model instance as Extensible Markup Language.

8. (Original) The method of Claim 7 wherein establishing a relationship further comprises:

activating plural contexts simultaneously to buffer data for output as a complete Document Object Model instance.

9. (Original) The method of Claim 8 wherein establishing a relationship further comprises:

creating a node for an output data; and

ensuring the correct cardinality of the created node.

10. (Original) The method of Claim 7 further comprising:

generating output data with an application;

calling a writer engine with the application;

providing the generated output data to the writer engine;

outputting data from a Document Object Model instance from the writer engine according to the Extensible Markup Language schema.

11. (Original) The method of Claim 10 wherein the application comprises a legacy computer system application.

12. (Original) The method of Claim 11 wherein the writer engine comprises an application run in the computer language of the legacy computer system application.

13. (Original) A system for outputting data from a Document Object Model as Extensible Markup Language, the system comprising:

a computer system having an application that outputs data; and

a writer engine loaded on the computer system and interfaced with the application, the writer engine having an Extensible Markup Language schema as a data file and the writer engine operable to write the output data in plural active contexts;

wherein the application calls the writer engine when the application outputs data, the writer engine operable to build a Document Object Model instance for output of the data in accordance with the Extensible Markup Language schema.

14. (Original) The system of Claim 13 wherein the writer engine populates a Document Object Model as a schema element aligned with the current one of the contexts by creating Extensible Markup Language tagged nodes down through the schema element of the output data if the schema element of the output data is a descendant of the current context.

15. (Original) The system of Claim 14 wherein the writer engine is further operable to determine a minimal mutual ancestor of the schema element and the current context and to traverse the Extensible Markup Language tagged nodes for the current context up to the minimal mutual ancestor and to create Extensible Markup Language tags for the schema element down from the mutual ancestor.

16. (Previously presented) The system of Claim 13 wherein the computer system comprises a legacy computer system.

17. (Original) The system of Claim 16 wherein the application comprises a legacy computer system application modified to output an Extensible Markup Language schema element with output data.

18. (Original) The system of Claim 17 wherein the writer engine is written in the code of the legacy computer system.

19. (Original) The system of Claim 18 wherein the code comprises COBOL.

20. (Original) A method for outputting data from a legacy computer system from a DOM instance as Extensible Markup Language, the method comprising:

modifying an application of the legacy computer system to output data having a schema element;

generating data from the modified application;

aligning the schema element and the current context;

writing the output data schema element to a current one of plural contexts of an Extensible Markup Language schema; and

populating a Document Object Model with the data to output an Extensible Markup Language instance.

21. (Original) The method of Claim 20 wherein aligning the schema element further comprises:

determining that the schema element is a descendant of the current context; and

creating the Extensible Markup Language tags down through the schema element.

22. (Original) The method of Claim 21 wherein aligning the schema element further comprises:

determining a minimal mutual ancestor of the schema element and the current context;

traversing the Extensible Markup Language tags for the current context up to the mutual ancestor; and

creating the Extensible Markup Language tags for the schema element down from the mutual ancestor.

23. (Previously presented) A method for modeling a legacy computer system comprising:

identifying incidents of applications of the legacy computer system that output data;

associating the incidents with an Extensible Markup Language schema;

defining a control flow graph of the output incidents;

creating a specification to modify the legacy computer system applications to provide output from a Document Object Model instance as Extensible Markup Language; and

automatically modifying the legacy computer system applications in accordance with the specification.

24. (Canceled).

25. (Previously presented) A system for modeling an output application of a legacy computer system comprising:

a modeling engine interfaced with the legacy computer system, the modeling engine operable to analyze an application loaded on the legacy computer system to identify incidents within the application that output data from the legacy computer system;

a control flow graph of the output operations within the applications, the control flow graph having plural nodes, each node associated with an output incident;

a graphical user interface in communication with the modeling engine, the graphical user interface operable to display the control flow graph and the incidents, wherein the graphical user interface maps the incidents of the applications with the control flow graph and an Extensible Markup Language schema; and

a code generation engine in communication with the graphical user interface and the legacy computer system, the code generation engine operable to modify legacy computer system application code to directly output data from a Document Object Model as Extensible Markup Language.